Fred Martin

Toy Projects Considered Harmful

It is valuable to improve upon the sample code we provide to students. But the deeper challenge is to create a context in which student programming matters.

Continuing the string of "Considered Harmful" essays that have appeared in *Communications*, I too owe a debt of gratitude to Edsger Dijkstra [2]. Chenglie Hu's "Technical Opinion" column, "Dateless Objects Considered Harmful," pointed out that when we introduce novice programmers to objects, then those objects had better exhibit key properties of "object-hood" [3].

The deeper issue, though, is that we are happy to have students spend semester after semester writing code that never sees the light of day. Students know their code matters only as much as they might find our assignments interesting, or as much as it counts toward their grades.

In contrast to this, I recently spent a week programming an online customer database and order-taking system for a small business that my wife and I operate. It had been a while since I spent any extended period of time writing code, and even longer since I wrote code that was going live.

My entire coding experience was shaped by this reality, and it had repercussions in my design process, choice of technology, and project schedule.

The overarching significance was emotional—I knew this code would underpin our business. It had to work, it had to work properly, and it had to be built quickly.

It struck me that many of my students might never have had this experience—the experience of writing code that matters.

If you are an academic, try to remember the last time you wrote code that went live or shipped. Code for classroom use, or even research use, may not count—it's more like the projects we assign to our students.

Coding for the Real World

Why the Web is now better than a book. When I first learned how to program, I used books. Later, as an undergraduate, I also learned by talking with peers. This was typical.

A few years ago, when I did a couple of small Web projects using PHP and MySQL, I bought a book. This time, I barely cracked it open—but not because I didn't need any documentation or resources. Instead, I Googled for help—and found more than I expected.

Besides being instantly available and free, online programming guides have a key feature that books do not: user comments. On each page documenting a function or feature, there is a batch of user comments (for example, see the home page for MySQL documentation at dev.mysql.com and the home page for PHP documentation at www.php.net). The user comments are often quite relevant—highlighting subtle interactions, or showing how to use a feature by example rather than by description. It strikes me that these combined technical manuals/user forums replace what in the past would be accomplished by talking to people around the lab.

Learning from sample code—or just using it. Since the objective of my project was to produce a functional Web commerce system, it made sense to be expedient and reuse or adapt whatever code I could find.

For example, I had previously adapted a Perl script, formmail.pl, to our site. My current project was being done in PHP, and I postulated there must be a formmail.php script. I downloaded it and perused it for my needs.

It is important to note that this sort of code reuse, for commercial purposes, is legitimate only if authors allow it. Fortunately, much code that is published on the Web includes generous licenses. For example, the formmail.pl script "may be used and modified free of charge by anyone" as long as its copyright notice and authorship attributions are preserved [5].

Our new warehouse company provided a PHP script that produced the necessary XML file for transferring daily shipment data. This script was valuable, both practically and for my own learning. It both demonstrated a structure for a customer/order database—which I adopted—and it illustrated coding techniques of which I had been previously unaware.

Consider two notes on this. First, writing code from scratch is often the wrong way to go about getting something done. If someone has already written, published, and is giving away code, it can be much quicker to use it than to write your own (for example, by using standard programming libraries).

Second, it's great to learn from others' code. Well-written code can both illustrate good form and provide a meaningful context for demonstrating ideas. I would often much rather see a short example program than read an abstract (though complete) specification.

How would student programming projects change if assignments were structured like "find some code on the Web that's close to what I'm asking and extend it to accomplish the assignment" versus "implement this functionality"?

On rapid prototyping. My project was an exercise in rapid prototyping. I had budgeted a week. I planned to do the design while writing the code. The software tools I used—which typically are not favored in academic settings—were key to this approach.

Interactive environments. The project used HTML, PHP, and MySQL, all of which are highly interactive. With HTML and PHP, there is no compile step: you reload your code in your browser to see what it does.

My own coding style is to write a few lines of code and test them. I mean this quite literally; I would write 3 to 10 lines before rerunning the code. At first, this was to test my understanding of what was going on. Is this stuff working as I think it is?

But later, 3 to 10 lines represented the size of a small operational chunk of progress. I saw no need to proceed to writing the next bit of code until the just-written bit was tested and operational.

Is this normal? For me, I can't imagine working any other way. But observing most of my students, this is deviant! Most students sit down and write one or several *pages* of code, then get it to compile, and then say, "it's done, but I still have to test it." Of course, they're nowhere near being done.

A GUI that helps you learn. I used a MySQL database, which includes both a command shell and a graphical front end (phpMyAdmin). Both let you design, build, and modify your database interactively. However, phpMyAdmin is more than your typical GUI, because it tells you what it's doing. Each time you use it, it generates a database query, and then displays it.

I generally do not like GUIs, and I now realize it's because they usually do the opposite: they accomplish some task by configuring something behind my back. Here, the expert knowledge built into the GUI teaches me while I am also getting something useful done.

On untyped languages. I know this is a sensitive issue, but I don't see how one cannot be "moved" by the experience of typing in PHP code and having it just work. Is that variable holding an integer? OK, sure. Is it a string? Sure. Is it holding a connection object? Of course. Is it holding a MySQL query result? Yes. What is that exactly? I don't care—I just know how to use it!

Yes—for big projects, one needs structured design principles and tools that help you validate your work.

But it is a powerful experience to write code that does complex things with a minimum of effort. We shouldn't protect our students from this feeling of power and expressiveness—instead, we should encourage them to be exhilarated by it.

I encourage students to be creative in reframing their problem based on their developing insights along the way.

Making It Real in the Classroom

The challenge, then, is having "making it real in the classroom" not be a contradiction in terms. I have found approaches that make classroom projects feel more real to students, including the following:

Students developing their own designs. There is always a balance between providing assignments that are focused on particular curricular topics and ones that are more open-ended. Our goal as teachers, though, is always to have our students become invested in the work.

Assignments that support student design can be particularly successful. In my department, a colleague developed a two-semester software engineering course based on computer gaming. First students design a game; then they implement it [1]. In our robotics course, students design a robot to perform a specified task; the task is clearly defined, but the solution space is wide and varied. In a course module developed in collaboration with a social scientist, students create working concept prototypes around the theme of sustainable development [4]. Ultimately, when students finish a class thinking, "I built something—this is mine," we have succeeded.

A benefit of design-based projects is that students are welcome to take advantage of whatever resources they can marshal, including using code located on the Internet. (As noncommercial users, even more code will be available to them.) By definition, their design is unique, and they will get further by leveraging published code and other work.

Encouraging rapid prototyping. Even if students are using traditional software tools, I encourage them to develop their code iteratively. In the classroom, I demonstrate ideas by writing 10 lines of C code, compiling it, and running it. Students help debug when something unexpected happens. I demonstrate debugging techniques like binary search to find problems. I want them to see that it's really helpful to have just a tiny bit of code running so that a particular concept or interaction can be understood.

In more traditional problem-solving assignments, I urge students to use rapid prototyping techniques to help them understand the ideas they are working with. In the design-based projects, rapid prototyping is even more important. Here, students are framing a problem and then designing its solution. The interaction between framing and solving is crucial. I encourage students to be creative in reframing their problem based on their developing insights along the way. When students realize they can't do what they initially set out to do, I ask them to target a more refined, limited goal.

A public performance. Nothing motivates work like a deadline, especially one that involves other people showing up to see what you did. Public events can take a variety of forms; I've organized elimination-style robot competitions and mini-symposia where each student/team gets a short presentation slot. My current favorite is the open-house-style event, where students set up their projects, including poster boards, and demonstrate them to visitors. This is best when student work involves some sort of artifact or program that can be shown doing something—making the event more than a poster session.

Providing a safe venue for risk. If students are developing a project that will be used by others who are depending on its timely, successful completion, there are real repercussions for failure, which ultimately will be borne by the supervising faculty. I'm really only willing to do this for projects that are part of my research plan (others may be more adventuresome).

The public presentation is a nice middle ground, for both students and faculty. Here, the main risk is social embarrassment—losing face in front of your peers because you obviously did not make a sufficient effort. This becomes a motivator for both students and faculty, as our joint learning venture is actually what is on display.

Really making it real. The best way, of course, is to bring student projects out of the classroom—that is, to have students develop something of value to people and organizations in the community.

The problem is difficult: the faculty's effort in developing a network of potential "customers"; eliciting "problem specifications"; framing these into something achievable in a semester or less; helping students deliver tested and reasonably debugged solutions; and maintenance after the fact.

My institution has small and large initiatives brewing. Small projects include having our students do writing projects for local nonprofit organizations, creating advertising and logos for restaurants and other small businesses, and developing novel assistive technology projects. Larger programs include a service-learning project across the Engineering school's curriculum and a institution-wide framework for community-university partnerships being led by one of our accomplished, senior faculty members.

With support from our institutions, junior and senior faculty members alike can make the necessary investment and create contexts for meaningful student work. C

References

1. Claypool, K. and Claypool, M. Teaching software engineering through game design. In *Proceedings of the Tenth Annual Conference on Invention and Technology in Computer Science Education (ITiCSE)* (Monte da Caparica, Portugal, June 2005).
2. Dijkstra, E.W.G. Go To statement considered harmful. *Commun. ACM 11*, 3 (Mar. 1968).
3. Hu, C. Dataless objects considered harmful. *Commun. ACM 48*, 2 (Feb. 2005).
4. Martin, F. and Kuhn, S. Computing in context: Integrating an embedded computing project into a course on ethical and societal issues. In *Proceedings of the 37th SIGCSE* (Houston, TX, 2006).
5. Wright, M. formmail.pl version 1.92; scriptarchive.com.

Fred Martin (fredm@cs.uml.edu)
is an assistant professor in the Department of Computer Science at the University of Massachusetts Lowell.